Homework #8
Due by Friday 9/7 11:55pm

Submission instructions:

1. Pay special attention to the style of your code. Indent your code correctly, choose meaningful names for your variables, define constants where needed, choose most suitable control statements, break down your solutions by defining functions, etc.
2. You should submit your homework in the NYU Classes system.
3. For this assignment, you should turn in 7 files:
   - Six '.cpp' files (for questions 1 - 6).
     Name these files: 'YourNetID_hw8_q1.cpp', 'YourNetID_hw8_q2.cpp', etc.
   - One '.pdf' file (for questions 7 - 9).
     Name this file 'YourNetID_hw8_q7to9.pdf'.

Question 1:

1. Implement the function: `int minInArray(int arr[], int arrSize)`
   This function is given `arr`, an array of integers, and its logical size, `arrSize`. When called, it returns the minimum value in `arr`.

2. Write a program that reads from the user a sequence of 20 integers (unnecessarily different from one another) into an array, and outputs the minimum value, and all the indices it appears in the array.

   Your program should interact with the user **exactly** as it shows in the following example:
   Please enter 20 integers separated by a space:
   14 5 12 5 6 14 5 12 14 12 14 6 8 7 5 136 9 2189 10 6
   The minimum value is 5, and it is located in the following indices: 1 3 6 14

   Note: You may want to define additional functions for your program to use.

Question 2:

A **palindrome** is a word, which reads the same backward or forward. For example, *noon*, *civic*, *radar*, *level*, *rotor*, *kayak*, *reviver*, *racecar*, *redder*, *madam*, and *refer* are all palindromes.

a. Implement a function:

    `bool isPalindrome(string str)`

    This function is given a string `str` containing a word, and returns `true` if and only if `str` is a palindrome.

b. Write a program that reads a word from the user and announces to the user if it is a palindrome or not.

    Your program should interact with the user **exactly** as it shows in the following example:

    Please enter a word: level

    level is a palindrome

Question 3:

Implement following functions:

a. `void reverseArray(int arr[], int arrSize)`

    That takes `arr`, an array of integers, and its size, `arrSize`. When called, it reorders the elements of the array to appear in a reverse order.

    For example, if `arr` is an array containing [1, 2, 3, 4], after calling `reverseArray`, `arr` will look like: [4, 3, 2, 1].

b. `void removeOdd(int arr[], int& arrSize)`

    That takes `arr`, an array of integers, and its size, `arrSize`. When called, the function alters `arr` so that the only numbers in it at the end are the even ones, which should remain in their original relative order.

    Additionally, the function updates `arrSize` so it contains the new logical size of the array after removing the odd numbers (note that `arrSize` is a parameter used both for input and output).

    For example, if `arr` is an array containing [1, 2, 3, 4], after calling `removeOdd`, `arr` will look like: [2, 4], and the parameter `arrSize` will update to 2. Notice the values in `arr[2]` and `arr[3]` are discarded.

c. `void splitParity(int arr[], int arrSize)`

    That takes `arr`, an array of integers, and its size, `arrSize`. When called, the function changes the order of numbers in `arr` so that all the odd numbers will appear first, and all the even numbers will appear last. Note that the inner order of the odd numbers and the inner order of the even numbers don't matter.

    For example, if `arr` is an array containing [1, 2, 3, 4], after calling `splitParity`, `arr` could look like: [3, 1, 2, 4].

Implementation requirements:

1. In all three functions, you are **not allowed** to use an auxiliary array (a temporary local array).
2. Pay attention to the running time of your functions. For each one of the functions above, an efficient implementation would run in a linear time (that is $\Theta(arrSize)$).

Note: You don't have to submit a `main` function for this question. You may use the following program to test your functions:

```cpp
#include <iostream>
using namespace std;

void printArray(int arr[], int arrSize);

int main() {
    int arr1[10] = {9, 2, 14, 12, -3};
    int arr1Size = 5;

    int arr2[10] = {21, 12, 6, 7, 14};
    int arr2Size = 5;

    int arr3[10] = {3, 6, 4, 1, 12};
    int arr3Size = 5;

    reverseArray(arr1, arr1Size);
    printArray(arr1, arr1Size);

    removeOdd(arr2, arr2Size);
    printArray(arr2, arr2Size);

    splitParity(arr3, arr3Size);
    printArray(arr3, arr3Size);

    return 0;
}

void printArray(int arr[], int arrSize){
    int i;

    for (i = 0; i < arrSize; i++) {
        cout<<arr[i]<<' ';
    }
    cout<<endl;
}
```

When running this program you should expect the following output (the output for `splitParity` could be different):

```
-3 12 14 2 9
12 6 14
1 3 6 4 12
```

Question 4:

Traditional password entry schemes are susceptible to "shoulder surfing" in which an attacker watches an unsuspecting user enter their password or PIN number and uses it later to gain access to the account. One way to combat this problem is with a randomized challenge-response system. In these systems the user enters different information every time, based on a secret in response to a randomly generated challenge.

Consider the following scheme, in which the password consists of a five-digit PIN number (00000 to 99999). Each digit is assigned a random number that is 1, 2, or 3. The user enters the random numbers that correspond to their PIN instead of their actual PIN numbers.

For example, consider an actual PIN number of 12345. To authenticate the user would be presented with a screen such as:

```
PIN:  0 1 2 3 4 5 6 7 8 9
NUM:  3 2 3 1 1 3 2 2 1 3
```

The user would enter 23113 instead of 12345. This doesn't divulge the password even if an attacker intercepts the entry because 23113 could correspond to other PIN numbers, such as 69440 or 70439.

The next time the user logs in, a different sequence of random numbers would be generated, such as:

```
PIN:  0 1 2 3 4 5 6 7 8 9
NUM:  1 1 2 3 1 2 2 3 3 3
```

Write a program to simulate the authentication process. Store an actual 5-digit PIN number in your program (make one up, and store it as a constant). The program should use an array to assign random numbers to the digits from 0 to 9. Output the random digits to the screen, input the response from the user, and output whether or not the user's response correctly matches the PIN number.

Assuming that the actual PIN number is 12345, your program should interact with the user **exactly** as it shows in the following examples (2 different executions of the program):

Please enter your PIN according to the following mapping:

PIN: 0 1 2 3 4 5 6 7 8 9

NUM: 3 2 3 1 1 3 2 2 1 3

23113

Your PIN is correct

Please enter your PIN according to the following mapping:

PIN: 0 1 2 3 4 5 6 7 8 9

NUM: 1 1 2 3 1 2 2 3 3 3

23113

Your PIN is not correct

Note: Think how to break down your implementation to functions.

Question 5:
Write a program that reads a person's name in the following format:
first name, then middle name or initial, and then last name.
The program then outputs the name in the following format:
Last_Name, First_Name Middle_Initial.

For example, the input
Mary Average User
should produce the output:
User, Mary A.

The input   **Mary A. User**
should also produce the output:
User, Mary A.

Note that your program should work the same and place a period after the middle initial even if the input did not contain a period.

Hint: You may want to use three string variables rather than one large string variable for the input. You may find it easier to **not** use `getline`.

Question 6:
Write a program that reads in a line of text and outputs the line with all the digits in all integer numbers replaced with 'x'.
Please enter a line of text:
My userID is john17 and my 4 digit pin is 1234 which is secret
My userID is john17 and my x digit pin is xxxx which is secret

Notes:
1. If a digits is part of a word, then the digit is not changed to an 'x'. For example, john17 is NOT changed to johnxx.
2. You may assume that the text entered by the user will contain only letters (upper case or lower case), digits and spaces.
3. Think how to break down your implementation to functions.

Question 7:

Each of 26 cards has a different letter of the alphabet on it. You pick one card at random. The value of the card is defined as 3 points, if the letter on it is a vowel, or 1 point if the letter is a consonant.

Let X = the value of the card picked.

a. Find the distribution of X
b. Find the expected value of X
c. Find the variance of X

Question 8:

You have two decks of 26 cards each. Each card in each of the two decks has a different letter of the alphabet on it. You pick at random one card from each of the two decks. The value of each card is defined as 3 points, if the letter on it is a vowel, or 1 point if the letter is a consonant.

Let X = the sum of the values of the two cards picked.

a. Find the distribution of X
b. Find the expected value of X in two ways:
   The first, by directly using the definition of the 'expected value of a random variable'.
   The second, by expressing X as a sum of two random variables

Question 9:

a) What is the expected sum of the numbers that appear on two dice, each biased so that a 3 comes up twice as often as each other number?
b) What is the expected sum of the numbers that appear when three fair dice are rolled?